In [1]:
```
%%capture
# This snipped of code will download the data file
!wget https://www.dropbox.com/s/ms5p2hqqhlaago7/replay_buffer.npy?dl=1
!mv replay_buffer.npy?dl=1 replay_buffer.npy
```

The code below solves the Bellman equation using TD with neural networks function approximation.

In [2]:
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Activation
import numpy as np
import matplotlib.pyplot as plt

γ = 0.95
batch_size = 128

replay_buffer = np.load('replay_buffer.npy')

replay_buffer
```

Out[2]:
```
array([[1.7544187 , 1.6899673 , 0.32488784, 0.35014135],
       [1.5905526 , 1.617272  , 0.39527917, 0.382326  ],
       [1.6724763 , 1.6786338 , 0.3575033 , 0.35488534],
       ...,
       [1.4697709 , 1.3415053 , 0.4629144 , 0.55566776],
       [2.2108552 , 2.2101867 , 0.20458762, 0.20471142],
       [1.322186  , 1.2165726 , 0.5720248 , 0.67565334]], dtype=float32)
```

In [3]:
```
model = Sequential([Dense(20, activation='relu'), Dense(1, activation='linear')])
model.compile(optimizer='Adam', loss='mse')
```

In [4]:
```
# This function will select a random sample of the replay buffer.
def sample_from_buffer():
    idx = np.random.choice(len(replay_buffer), batch_size, replace=False) # A vector of random indexes,
    # where the indexes can vary from 0 to len(replay_buffer).
    # The size of the sample is 'batch_size'

    mini_batch = replay_buffer[idx]
    Dt, D_tp1, π_t, π_tp1 = np.split(mini_batch, 4, axis=1)
    R = γ * π_tp1 * D_tp1
    return Dt, R, D_tp1


# This function implements one step of the TD update
def update():
    D, R, D_tp1 = sample_from_buffer() # get a sample from the replay buffer
    V_tp1 = model.predict(D_tp1) # Evaluate the network at y_tp1
    TD_target = R + γ * V_tp1
    model.fit(D, TD_target, validation_split=0.1, verbose=False, batch_size=batch_size)
```

In [9]:
```
for iteration in range(10000):
    update()
```

In [10]:
```
# Show your results
Dt, D_tp1, π_t, π_tp1 = np.split(replay_buffer, 4, axis=1)
V = model.predict(Dt)
Pt = V / π_t
plt.figure(figsize=(7,6), dpi=100)
plt.scatter(Dt, Pt, s=1)
plt.show()
plt.pause(1e-6)
# P.S: you figure should be similar to this one: https://www.dropbox.com/s/6992cmfatiu3wkj/PD.png?dl=0
```